Powershell mis see on?

Kui PowerShell käivitada, siis avaneb esmalt käsurida, kuid PowerShell on märksa enamat kui lihtsalt käsurida. PowerShell on:

- Käsurida
- Hulk käsurea vahendeid, käske või käsklusi (cmdlets) Skriptikeel

Kokkuvõtvalt võiks öelda, et PowerShell on automatiseerimismootor, mis võimaldab kaugligipääsu erinevatele arvutitele, asünkroonset protsessimist ning tänu WMI (Windows Management Instrumentation), COM- ja .NET komponentidele lihtsale kasutusvõimalusele on see väga hõlpsasti laiendatav.

PowerShell v1 avaldati 14. novembril 2006 aastal. PowerShell v2 arendati välja koos Windows 7 ja Windows Server 2008 R2'ga ning on nende vaikimisi nende operatsioonisüsteemide koosseisus. Windows XP, Windows Server 2003, Windows Vista ja Windows Server 2008 peale tuleb PowerShell v2 eraldi paigaldada. Koos Windows 10 ja Windows Server 2016 arendamisega on PowerShell jõudnud versioonini 5.1 ning see on saadaval eraldi paigalduspaketina Windows 8.1, Windows 7, Windows Server 2012, Windows Server 2012 R2, Windows Server 2008 ja Windows Server 2008 R2 jaoks kuid mitte enam Windows XP, Windows Server 2003 ja Windows Vista jaoks.

Alates augustist 2016 on Powershell avatud lähtekoodiga (Open Source) ning PowerShell v6 Alpha versioonid on saadaval ka Linux'ile ja macOS'ile. https://github.com/PowerShell/PowerShell

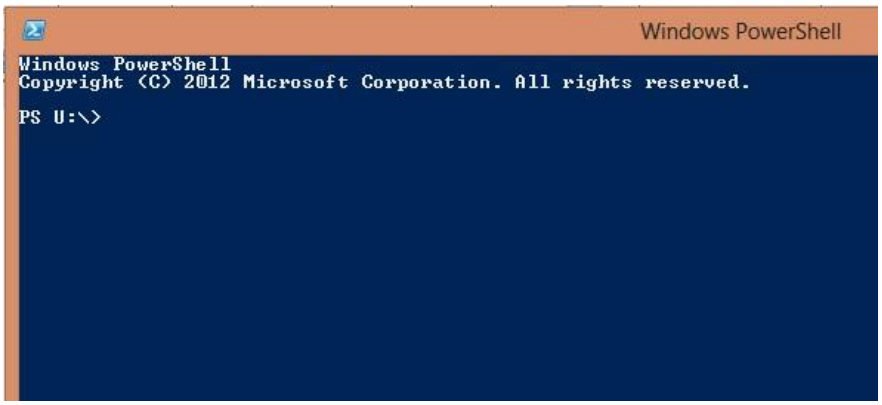
Kuna PowerShell on hõlpsasti laiendatav, siis on olemas PowerShelli laiendusmoodulid väga mitmete tarkvarapakettide jaoks nagu näiteks Exchange Server, SharePoint Server ja isegi SQL server. Samuti on PowerShellile laiendusmooduleid kirjutanud oma toodete käsurealt ja skriptide abil konfigureerimiseks sellised tarkvaratootjad nagu näiteks VMWare, Citrix, Cisco ja Quest. PowerShelli lisamoodulite valik täieneb iga päevaga ja nende koondamiseks on loodud eraldi keskkond https://www.powershellgallery.com/

Käsurida on eksisteerinud nii Windowsil kui ka teistel operatsioonisüsteemidel sisuliselt algusest peale. PowerShelli teeb võrreldes teiste käsuridadega eriliseks asjaolu, et tegemist on objekt orienteeritud käsureaga. See tähendab, et erinevate käskude sisendid ja väljundid ei ole mitte lihtsad tekstistringid vaid objektid.

Rohkem infot leiad aadressilt: https://msdn.microsoft.com/en-us/powershell

Käsurida

Klikkides PowerShelli ikoonil  avaneb PowerShelli käsurida:



Käsureale võib hakata koheselt kirjutama käske ja neid käivitades tagastatakse koheselt tulemus.

Näiteks tipi käsureale järgnev käsk ja käivita see klahvivajutusega <Enter>:

```
Get-Help
```

Käsureale käske kirjutades ei pea käske terves pikkuses välja kirjutama. Pooleldi kirjutatud käskude lõpetamiseks kasuta klahvi <Tab>.

```
Get-He <Tab>
```

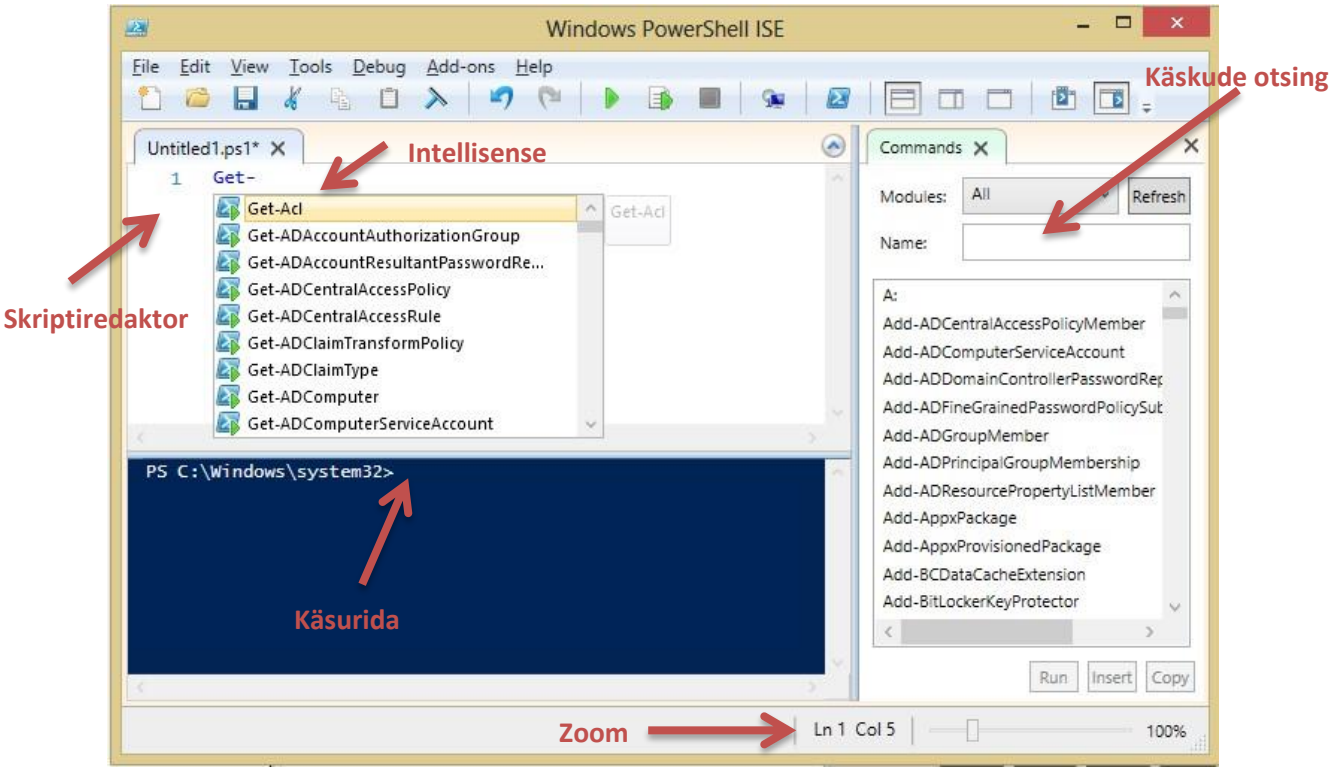
Kui juba tipitud teksti lõpetamiseks on mitu võimalust võib <Tab> klahvi vajutada korduvalt

```
Tipi: Get-H <Tab>
Tagastatakse: Get-Help
Vajuta: <Tab>
Tagastatakse: Get-History
Vajuta: <Tab>
Tagastatakse: Get-Host
Vajuta: <Tab>
Tagastatakse: Get-HotFix
Vajuta: <Tab>
Tagastatakse: Get-Help
```

<Tab> klahv toimib ka käsu parameetrite otsimiseks/lõpetamiseks

```
Tipi: Get-He <Tab> -<Tab>
Tagastatakse: Get-Help –Name
```

Lisaks tavalisele käsureale paigaldatakse koos PowerShelli käsureaga arvutisse ka interaktiivne PowerShelli konsool PowerShell ISE (Selle leiad Windowsi otsingu abil)



Interaktiivse konsooli abil on mugav kirjutada skripte ja PowerShell ISE abil on võimalik skripte ka siluda (Debug). Kui PowerShell ISE konsooli avades on skriptiredaktori paan varjatud, siis tuleb see käsitsi sisse lülitada, valides „View" menüüst „Show Script Pane" või vajutades klaviatuuril Ctrl+R. PowerShell ISE konsooli on võimalik ka lisadega (Add-ons) täiendada. Üks eelinstalleeritud lisa on Commands, mis võimaldab kiirelt PowerShelli käskude kohta abiinfot otsida. Lisasid on võimalik alla laadida veebist (lingi leiad PowerShell ISE Add-ons menüüst) või ise kirjutada.

Käsurea vahendid või käsklused (Cmdlets)

PowerShell v2 koosseisus on kokku umbes 230 nö. käsku või käsurea vahendit. PowerShell v3 koosseisus on käsurea vahendite hulka oluliselt kasvatatud ja nüüd on neid baasvarustuses kokku üle 2400. Hilisemates versioonides on see number veelgi suurem. Neid käsureavahendeid nimetatakse *cmdlets*. Käsk koosneb kahest sõnast, mis on eraldatud sidekriipsuga – näiteks `Get-Process`. Käsu esimene osa on tegusõna ja teine nimisõna (mis peab olema unikaalne). Tõsi ei ole kõikide käskude tegusõna grammatilises mõttes päris tegusõna näiteks `New`, mida tuleks tegevuse saamiseks pisut täiendada „create a new". Seega nimetagem siis käskude esimest poolt pigem tegevuseks kui tegusõnaks ☺. Selline unifitseeritud skeem teeb aga käskude meelde jätmise lihtsaks ja sisu loogiliselt arusaadavaks. Näitena toodud käsk `Get-Process` tagastab informatsiooni hetkel töötavate protsesside kohta kas siis lokaalses või mõnes kaugarvutis.

Kõiki PowerShellis kasutatavaid tegusõnu saate pärida käsuga

```
Get-Verb
```

Kasutatavate käskude nimekirja saate pärida käsuga

```
Get-Command
```

Üldised käsklused (Utility cmdlets)

Üldised käsklused toimivad nö. liimina, mille abil on võimalik teisi käsklusi omavahel siduda. Nende abil on võimalik andmeid filtreerida, sortida, võrrelda ja grupeerida või isegi uusi objekte koostada. Sellised käsklused sisaldavad sõna `Object`. Kui paljude skriptikeelte juures on sellised funktsioonid vaja ise koostada, siis PowerShellil on need koheselt olemas.

Selliste käskude nimekirja saate pärida sisestades käsureale

```
Get-Command *-object
```

Tulemuseks saate 9 käsklust.

Nagu näha viimati kasutatud käsust, on võimalik üksikule käsklusele lisada parameetreid ja selle abil käskluse toimetamist juhtida. Iseenesest on kasutasite viimati täpselt sama `Get-Command` käsklust nagu varem, aga tulemuseks saite oluliselt lühema nimekirja. Käsklusele lisatud parameeter `*object` tähendab, et filtreeri käskluse kõikidest tulemustest välja ainult need, mis lõppevad „object" (* tähendab, et algus võib olla suvaline).

Tutvuge eelpool saadud üldiste käskude abiinfoga ja tehke endale selgeks, milleks neid kasutada saab. PowerShelli käskude kohta saate abiinfot pärida käsklusega `Get-Help`.

Vaikimisi sisaldab PowerShelli installatsioon käskluste kohta piiratud hulgal abiinfot. Abiinfo täiendamiseks võite käivitada käskluse `Update-Help`, mis laeb internetist alla kõige viimase redaktsiooni täielikust abiinfost või kui kogu infot alla laadida ei taha, siis võite lihtsalt `Get-Help` käsule lisada parameetri `-Online`. Ehk siis

```
Get-Help <käsklus mille kohta soovid infot>
```

või

```
Get-Help <käsklus mille kohta soovid infot> -Online
```

Väike meeldetuletus: Kasuta <Tab> klahvi ja seda ka käskluse parameetrite korral.

Osade käskluste jaoks on kasutusel ka aliased. Näiteks võib `Select-Object` asemel kasutada lihtsalt `select` või siis `Where-Object` asemel kasutada lihtsalt `where`. Aliaste kohta leiate ka informatsiooni käskluse abiinfost või siis käsklusega `Get-Alias`.

PowerShelli süntaksist

PowerShelli käsurea süntaksi põhimõtetega tutvumiseks teeme läbi ühe näite ja kasutame selles juba eelmisest punktist tuttavaid käskluseid `Where-Object`, `Select-Object` ja `Sort-Object`.

Oletame, et te tahate kiiresti teada saada millised on teie arvuti PCI siinile ja USB portidesse ühendatud seadmete installeeritud draiverite versioonid. Loomulikult võite te avada Device Manager'i ja ükshaaval kõik seadmed üle käia ning draiveri versioonid üles märkida. Kindlasti ei ole aga selline tegevus kiire lahendus. Soovitud andmete kiiresti teada saamiseks käivitage käsklus

```
Get-WmiObject –Class Win32_PnpSignedDriver
```

Ilmselgelt on käskluse poolt tagastatud informatsiooni liiga palju. Antud käskluse tulemusena tagastati teile üks objekt iga draiveri kohta, mis on arvutisse installeeritud. Nagu juba eespool mainitud, on PowerShelli näol tegemist objekt orienteeritud käsureaga, ehk siis käskluste sisendid ja väljundid on objektid mitte lihtne tekst. Kui olete objekt orienteeritud programmeerimisega kokku puutunud, siis kindlasti teate, et igal objektil on olemas hulk omadusi (properties) ja tegevusi või meetodeid (methods). Samuti on ka PowerShelli käskluste poolt väljastatud objektidel. See nimekiri mis viimase käskluse poolt väljastati ongi selle käskluse poolt väljastatud objektide omaduste väärtuste nimekiri.

Objekti omaduste või meetodite poole pöördumiseks omistatakse objekt muutujale, kirjutatakse muutuja järele punkt ja omaduse või meetodi nimi.

Muutujad

Muutujate nimed algavad PowerShellis alati $ märgiga ja muutujaid on võimalik erinevate käskluste juures korduvalt kasutada. Muutujatele saab omistada nii teksti, numbreid kui ka terveid objekte või koguni objektide massiive. Näiteks

```
$tekst = „Vastus on „
$a = 5
$b = 6
$c = $a + $b
$tekst + $c
```

Või näiteks eelnevalt tagastatud draiverite objektide arvu teada saamiseks võime moodustada tagastatud objektidest uue objekti ja omistada selle muutujale ning seejärel küsida uue muutuja liikmete arvu.

```
$draiverid = Get-WmiObject –Class Win32_PnpSignedDriver
$draiverid.count
```

Kuna me seda uut muutujat hiljem kasutada ei taha võime kasutada ka lihtsustatud konstruktsiooni, mis püsivat muutujat ei tekitagi.

```
(Get-WmiObject –Class Win32_PnpSignedDriver).count
```

Mitu objekti tagastati?

Märkus!
Kui on huvi teada saada millistest meetoditest ja omadustest käskluse poolt väljastatud objekt koosneb, siis võite suvalise käskluse väljundi suunata käskluse Get-Member sisendisse. Tulemuseks saate nimekirja konkreetse objekti kohta.

Tagastatud objektide DeviceID väljal on täpne identifikaator, millise riistvarakomponendi jaoks antud draiver mõeldud on. PCI siinil olevate seadmete korral algab see identifikaator tähtedega PCI ja USB seadmete korral tähtedega USB. Seega täiendame pisut eelmist käsklust

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*“}
```

Mitu objekti nüüd tagastati?

Eelnevas käsus on kasutatud võrdlusoperaatorit –like. Teiste võrdlusoperaatorite kohta loe lisaks Powershelli abiinfost käsuga

```
Get-Help about_Comparison_Operators
```

Toru (pipeline)

Viimati käivitatud käsuga käivitati tegelikult kaks käsklust. Esimese käskluse `Get-WmiObject` väljund (väljastatud objektid) kanaliseeriti (piped) `Where-Object` käskluse sisendisse filtreerimiseks. Käskluste kokku kanaliseerimiseks kasutatakse käsureal käskluste vahel märki |. Sellist kanaliseeritud käskluste kogumit nimetatakse toruks (pipeline). Torus liigub korraga üks objekt ja seda objekti on järgneva käskluse juures võimalik kasutada spetsiaalse $_ muutuja abil ehk siis esimese käskluse väljundist tulnud objekt omistatakse automaatselt muutujale $_ ja seda on võimalik järgnevas käskluses kasutada nagu iga teist muutujat.

Skriptiplokk (script block)

Viimases täiendatud käsus on kasutusel veel üks PoweShelli konstruktsioon. Koodilõiku, mis jääb {} vahele nimetatakse skriptiplokiks. Skriptiplokk on käskude või avaldiste kogum, mida on võimalik kasutada kui ühte üksust. Skriptiplokk võib ka aktsepteerida argumente ja tagastada väärtuseid. Antud juhul võrreldakse skriptiplokis, kas torus liikuva objekti (mis on omistatud muutujale $_) omadus DeviceID algab tekstiga PCI või mitte. Kui algab, siis lastakse objekt edasi kui ei, siis filtreeritakse välja.

Olgugi, et selliselt filtreerides oleme saanud tagastatavaid objekte oluliselt vähemaks on jätkuvalt tagastatavat informatsiooni liiga palju. Täiendame veel pisut oma käsku

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion
```

Viimase täiendusega suuname tulemused veel torus edasi ja sedapuhku `Select-Object` käsklusele ning näitame ära, milliseid sisendobjekti omadusi me näha tahame. Sel korral on tulemused juba üsna hoomatavad ja peaaegu võime tulemusega rahul olla.

Märkus!
Kuna osad objektid tagastavad vaikimisi ekraanile kuvamiseks ainult piiratud hulga omadusi, siis kõikide objekti omaduste kuvamiseks võib kasutada käsklust `Select-Object *`

Viimase käsurea väljundi juures on häda veel selles, et tulemused on väga kaootilises järjekorras ja kirjelduse tulp on liialt kitsas (osade kirjelduste tekst lõigatakse lõpust ära). Ka nende probleemide vastu on lihtne lahendus

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion | Sort-Object Description | Format-Table –AutoSize
```

Mida viimased muudatused tegid?

Uuri, mida teeb käsklus Format-Table?

Tutvu ka teiste väljundformaati muutvate käsklustega

```
Get-Command format* -Module Microsoft.Powershell.Utility
```

Mida teeb parameeter –Module?

Kuidas saab teda, millised moodulid üldse konkreetses arvutis saadaval on?

Tulemuste kirjutamine faili

Powershelli käskluste väljundeid saab suunata ka faili. Üks lihtne viis väljundi faili suunamiseks on see, kui anda väljund toru abil edasi käsklusele Out-File. See käsklus kirjutab faili täpselt sellise väljundi, nagu muidu kuvatakse ekraanile.

Proovi, millise faili saad, kui lisad viimasele WMI päringu käsule lõppu:

```
| Out-File „C:\Mingi kaust\FailiNimi.txt"
```

Selliselt toimides on tulemusi küll lihtne faili suunata, aga kindlasti tuleb tähele panna, kas käskluste väljund suunatakse faili enne või peale Format- käskluste kasutamist. Enne tulemuste faili suunamist võib Format- käskluste asemel kasutada ka ConverTo- käskluseid:

ConvertTo-Csv, ConvertTo-Html, ConverTo-Json, ConvertTo-Xml

Võrdle järgnevate käskluste väljundeid:

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion | Sort-Object Description | Format-Table | Out-File
C:\Mingi kaust\FailiNimi.txt
```

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion | Sort-Object Description | ConvertTo-Csv | Out-File
C:\Mingi kaust\FailiNimi.csv
```

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion | Sort-Object Description | ConvertTo-Html | Out-File
C:\Mingi kaust\FailiNimi.html
```

```
Get-WmiObject –Class Win32_PnpSignedDriver | Where-Object
{$_.DeviceID –like „PCI*"} | Select-Object Description,
DriverVersion | Sort-Object Description | ConvertTo-Json | Out-File
C:\Mingi kaust\FailiNimi.json
```

Proovi ka Out-File käskluse asemel kasutada Out-GridView. Selle käskluse ees ära Format- ja ConvertTo- käskluseid kasuta.

Millise tulemuse saad?